Base de datos y visor web: clonismo comercial en millas de oro

MÁSTER EN GEOINFORMACIÓN 2019/2020

Autor: Noel Carrión Matías

Tutor académico: Meritxell Gisbert Traveria

Tutor de la entidad: Marc Vilà Recio

máster en geoinformación aplicaciones y gestión

ICGC Institut Cartogràfic i Geològic de Catalunya

UAB Universitat Autònoma de Barcelona

SOLUCIONS geoinformació i territori

Contenido

Resumen:

Ante la petición de almacenar y visualizar los datos recogidos en las millas de oro, de diferentes municipios catalanes, se ha desarrollado una base de datos en lenguaje *SQL* y un visor web, con una base de librería Leaflet, para cumplir los objetivos. El proyecto actual está estructurado con la metodología y exposición de resultados para cada uno de los apartados.

Palabras clave:

Milla de oro, base de datos y visor web.

Resum:

Davant la petició d'emmagatzemar i visualitzar les dades recollides en les milla d'or, de diferents municipis catalans, s'ha desenvolupat una base de dades en llenguatge SQL i un visor web, amb una base de llibreria Leaflet, per a complir els objectius. El projecte actual està estructurat amb la metodologia i exposició de resultats per a cadascun dels apartats.

Paraules clau:

Milla d'or, base de dades i visor web.

Abstract:

In response to the request to store and display the data collected in the golden miles, of different Catalan municipalities, a database in SQL language and a web viewer have been developed, with a library base Leaflet, to comply the goals. The current project is structured with the methodology and presentation of results for each of the sections.

Key words:

Mile of gold, database and finder web.

1 Introducción

1.1 Antecedentes y estado de la cuestión

Para el presente proyecto es necesario visualizar otros similares, ya sean realizados en anteriores ediciones del master en Geoinformación o de origen externo, para ver los posibles resultados.

En cuanto a la metodología, en las ediciones anteriores del máster en Geoinformación, se han realizado diferentes proyectos donde se crea una base de datos y una posterior visualización mediante su aplicación. Un ejemplo puede ser *Creació d'un visor web per la visualització de dades d'interferometria* del ICGC (Kevin González, 2018) o *Creació d'una xarxa de rutes ciclista: Topologia, visualització i càlcul dels camins òptims* (Pau Estruch, 2019). Estos ejemplos crearon una base de datos con *PostgreSQL*, cuyas tablas también tenían geometría, para posteriormente representarlas en un visor o conectarlas a *QGIS*. Por lo tanto, usan metodología y resultados parecidos, pero con un tema totalmente diferente.

Desde una perspectiva relacionada con el objeto principal, la publicación de comercios en una aplicación, hay un caso reciente a nivel español. El consorcio TIC de Mallorca ha creado una aplicación web con un visor, donde se muestran los comercios abiertos durante el estado de alarma por el Covid-19. No obstante, utiliza código de Google Maps, pero, salvando las distancias, la simbolización y el resultado es similar (TIC Mallorca, 2020).

Por último, solamente queda comentar que en el máster en Geoinformación Solucions Geogràfiques, la empresa en la que se desarrolla el presente proyecto, ha tenido poca presencia participativa. Uno de los proyectos realizados conjuntamente es *Millora del visor d'espais VAT* (Eva Sivillà, 2018). El objetivo de este, que prácticamente todo se realizó con la plataforma *CARTO,* es actualizar el visor que permite elaborar bases de datos y mapas interactivos sin necesidad de otros archivos. Sin embargo, el resultado visual es muy parecido al del presente proyecto, un visor con elementos categorizados, un *plugin* en común y un filtro. También se han realizado una aplicación para la recogida de

datos de campo (*Geosolución.* Roger Codina, 2019) o una aplicación web para la gestión de una explotación vitivinícola de Judit Mujal, en 2015.

En el aspecto más técnico de los proyectos mencionados, se han utilizado diferentes lenguajes de programación y librerías de mapas, a diferencia del proyecto de Eva Sivillà, como se ha mencionado anteriormente. El lenguaje para la creación de las bases de datos, se ha generalizado el código *SQL.* Para el diseño, el lenguaje CSS es el esencial y, en la elaboración de los visores, se ha utilizado el propio para diseñar páginas de Internet (*HTML5*). En cuanto a lenguaje PHP, solamente se ha aplicado en el proyecto de Pau Estruch.

En el caso de las librerías de mapas ya existe una mayor diversificación. Así pues, se ha utilizado *MapBox* (Kevin González, 2018), Leaflet (Pau Estruch, 2019) y Google (TIC Mallorca, 2020).

1.2 Marco institucional

Solucions Geogràfiques SCCL es una cooperativa que surgió en 2011 con el objetivo de dar a conocer el papel del geógrafo/a y poner en contacto a estos, más allá del ámbito académico. Es a partir del 2012-2013 en que crean su marca propia y su página web para comenzar a realizar proyectos para diversos clientes.

La temática en la que trabajan se divide en 3 áreas de trabajo:

- Consultora territorial: realización de planes o estudios de ordenación territorial, movilidad, medio ambiente y/o paisajismo.
- Espacios VAT: poner en valor aspectos territoriales, diagnosis, planes y proyectos.
- Geoinformación: gestión y dotación de valor a datos, visores cartográficos, sistemas de información propios o estudios de ubicación y su entorno.

Solucions cuenta con gran experiencia en el ámbito de la geoinformación, con proyectos como el del arbolado viario en Calafell o un visor con la potencialidad de explotación de huertos urbanos en las terrazas de Barcelona.

El proyecto de clonismo comercial “Clonatge Comercial 2019”, base del presente proyecto, surge con el objetivo de analizar los comercios de las millas de oro de diferentes municipios catalanes. La similitud en municipios es una problemática que se remonta al siglo pasado, pero en el caso de los comercios es debido a la expansión de diferentes cadenas comerciales. Estas concentran la gran mayoría de locales en las millas de oro, dejando poco espacio para el comercio tradicional o local. *Solucions Geogràfiques SCCL* ha participado en dicho proyecto en la fase de trabajo de campo.

1.3 Definiciones

- Millas de oro: tramo de calle o calles de un municipio donde se concentra una gran actividad económica.
- Clonismo comercial: similitud de los comercios según sus características comerciales.
- Leaflet: es una librería JavaScript en código abierto para la creación de mapas interactivos.
- Mapa base: mapa de referencia donde superponer y mostrar información ofreciendo un contexto geográfico.
- SQL: lenguaje de programación diseñado para la administración y gestión de datos.

2 Objetivos, planificación y estructura del proyecto

2.1 Objetivos generales

1. Construir una base de datos que contenga todos los datos comerciales de 2018-2019 registrados por Solucions Geogràfiques en el proyecto *“Clonatge Comercial 2019”.*
2. Generar un visor web en el que se geolocalicen los establecimientos comerciales, y su información asociada más destacada, con los que ha trabajado Solucions en el proyecto “Clonatge Comercial 2019”.

2.2 Objetivos específicos

Con tal de conseguir los objetivos generales, es necesario completar los específicos que son los siguientes:

- Objetivo general 1:
    - Generar el modelo conceptual para la estructuración de la base de datos.
    - Diseñar el modelo lógico para estructurar la base de datos.
    - Trasladar el modelo lógico *a PostgreSQL* e integrar todos los datos de los comercios.
    - Trasladar el modelo lógico al servidor de Solucions Geogràfiques.
    - Conectar la base de datos con el programa *QGIS.*
    - Ejecutar ediciones a través de *QGIS.*
- Objetivo general 2:
    - Determinar los requerimientos técnicos y casos de usos del visor.
    - Programación del visor web y conexión de la base de datos en él.
    - Enlazar la base de datos con el visor web.

2.3 Estructura y planificación del proyecto

Teniendo en cuenta los objetivos definidos anteriormente a cumplir se ha estructurado el proyecto en 6 partes diferentes que forman los bloques de base de datos y visor web.

Nº tarea	Nombre	Descripción
1	Modelización	Generar el modelo conceptual, lógico y URM
2	Carga de datos	Crear la base de datos en PostgreSQL
3	Conexión a QGIS y servidor web	Conectar la base de datos realizada con el QGIS y al servidor web
4	Diseño funcional e interfaz	Definir el aspecto y las funcionalidades del visor deseadas
5	Especificaciones técnicas y casos de uso	Elaborar las especificaciones técnicas y los casos de uso
6	Solución técnica	Introducir el código del visor

Tabla 1- Información de las tareas realizadas en el proyecto. Elaboración propia.

A continuación, se muestra la tabla adaptada a la duración en semanas del proyecto, teniendo en cuenta que tiene una duración máxima de 150 horas. En este caso, el periodo de las prácticas colaborativas comienza el 14 de abril y finaliza el 30 de junio.

Tareas	Abril			Mayo					Junio			
	Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Sem 6	Sem 7	Sem 8	Sem 9	Sem 10	Sem 11	Sem 12
1												
2												
3												
4												
5												
6												

Tabla 2- Gráfico de Gantt del proyecto. Elaboración propia.

3 Metodología

El presente proyecto cuenta con 2 partes principales y conectadas entre ellas (*Ilustración 1*):

1. La primera parte consta del diseño y generación de una base de datos en *PostgreSQL*. En esta se almacenarán todos los datos de los comercios analizados en diferentes municipios catalanes. Para su creación será necesario realizar un modelo conceptual y uno lógico. Será a partir de este segundo que se generará un script para importar la estructura diseñada al programa *PostgreSQL*.
2. El visor web es la siguiente fase. Se puede comenzar a diseñar antes de la finalización de la base de datos, pero, sin embargo, depende de ella para su correcto funcionamiento. Debido a su sencillez no tendrá más allá de los *plugin* de un buscador y filtros.

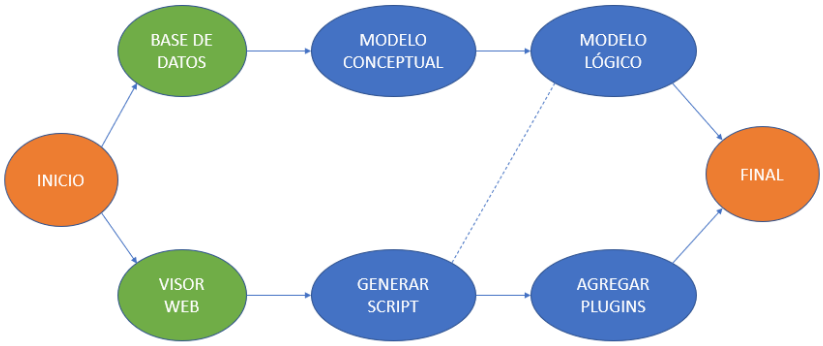


Ilustración 1- Esquema de las fases del proyecto. Elaboración propia.

3.1 Base de datos

Los datos originales para crear la base de datos fueron ofrecidos por la empresa en una hoja de cálculo. En total son 2.812 comercios analizados, con distintas categorías como son el tipo de actividad o el sector comercial. Inicialmente, había 27 campos para cada comercio, pero debido a la no homogenización en la toma de datos durante el trabajo de campo, se corrigieron diversos aspectos.

Sin embargo, a causa de la complejidad y el deseo de crear una base de datos y visor sencillo, se eliminaron varios campos, dejando finalmente 17. Todos estos cambios hicieron que el proyecto se ralentizara una semana (*Tabla 3*).

Fid	E
Id	E
Id_okk	C
Nom_estab	C
Sit_empr	M
Local_buit	E
Id_sec	E
Nom sector	E
Id_gro	E
Nom grup	E
Id_act	E
Nom activitat	E
X UTM	E
Y UTM	E
Clonisme	C

Class. Activitats milla or	C
Galeries	C
Carrer	C
Número	C
Municipi	C
Comarca	C
Observacions	C
Id_foto	E
Foto	C
Link foto	E
Link foto 2	E
Fecha	C
X GEO	C
Y GEO	C

E (Eliminado)	C (Correcto)	M (Modificado)

Tabla 3- Índice de columnas, la modificación y leyenda. Elaboración propia.

3.1.1 Modelo conceptual

En este apartado se explica cómo se ha realizado el diseño conceptual basado en un modelo de entidad-relación a través del programa *Open Model Sphere*. Este lo forman 3 tablas diferentes: ESTABLIMENT, CARRER y MILLA_OR. Entre ellas se ha establecido una relación de 1-N o 1-1, dependiendo la cardinalidad. A cada una se le han agregado los diferentes campos que las forman, al igual que un nombre, un nombre físico y una clave primaria (PK). Al principio se había diseñado un modelo con muchas más tablas, pero con el objetivo de sencillez, se eliminaron o reagruparon (Ilustración 3).

3.1.2 Modelo lógico

Para el modelo lógico, se ha seguido utilizando el programa *Open Model Sphere*. Traspasar de un modelo a otro se hace a través de un simple paso, en el cual se transforma automáticamente. Aún la sencillez del cambio, se hace un control de calidad de las tablas por si se ha generado algún error.

Una vez obtenido el modelo lógico, se pasan a mostrar los nombres físicos de los diferentes elementos y se generan las claves foráneas (FK). De este modo,

se tienen los campos identificadores de otras tablas o claves primarias (PK) en la tabla de ESTABLIMENT.

A continuación, se realiza de nuevo un control de calidad para que todos los elementos tengan su nombre físico y se hayan generado correctamente las claves foráneas (Ilustración 4).

3.1.3 Modelo UML

Finalmente, se crea un modelo UML, basándose en los modelos anteriores, desde el programa *Enterprise Architect.* Este paso generará un *script SQL* para crear las tablas y las relaciones entre ellas, de manera automática, en el programa *PostgreSQL* (Ilustración 5).

3.1.4 Carga de datos

La carga de datos se realiza parcialmente en el programa *PostgreSQL* con lenguaje *SQL.*

Primero de todo, es necesario crear una nueva data-base y un esquema dentro del programa. Una vez generados estos elementos, se selecciona la opción *SQL* para ejecutar el *script* obtenido en el apartado anterior. Este *script* creará todas las tablas y claves foráneas tal cual estaban en el modelo lógico. Del mismo modo, se pueden generar las tablas manualmente, ya sea a través de un script o con las opciones semiautomáticas que ofrece el programa.

Con las tablas ya creadas en *PostgreSQL* la opción más rápida y sencilla para la carga de datos es importarlos desde un archivo. Este paso se puede hacer de manera automática clicando sobre la tabla, con el botón derecho, o importando desde un archivo *.csv* o *.txt*. Este proceso también se podría hacer manualmente, pero, teniendo en cuenta que son más de 2000 comercios, esta metodología no sería rentable.

Ya con todos los campos rellenados, solamente falta crear la columna de geometría en la tabla de ESTABLIMENT. Esta se ha realizado directamente en el servidor de la empresa, debido a problemas con el *plugin PostGIS* de *PostgreSQL*. Así pues, aunque sin esta columna, se genera un *backup* de la

base de datos completa. Fruto de eso se genera un *script SQL* que se ejecuta en la consola *SQL* del servidor.

3.1.5 Importación al servicio web

El último paso de generación de la base de datos tiene como objetivo la importación de esta al servidor web de la empresa. Antes de importar, pero, es necesario crear una base de datos y un esquema en el servidor. Este paso es idéntico al realizado en *PostgreSQL*. Una vez generada esta estructura se procede a cargar el *script* del *backup* del paso 3.1.3. Para ello, solamente es necesario abrir la consola *SQL*.

La importación de la base de datos en el servidor termina con la creación del campo geometría, ya que como se ha comentado, no se podía generar en el *PostgreSQL* local. Para generar esta columna, que tiene el objetivo de almacenar la información espacial de la tabla para poderse representar, se ejecuta el siguiente *script*:

ALTER TABLE nombre_tabla ADD COLUMN nombre_columna GEOMETRY;

Y posteriormente el que se muestra a continuación para cargar los datos:

UPDATE nombre_tabla SET nombre_columna = St_MakePoint (campo_longitud, campo_latitud);

Para comprobar que todo este procedimiento se ha realizado correctamente, se conecta dicha base de datos a *QGIS* (*Ilustración 2*).

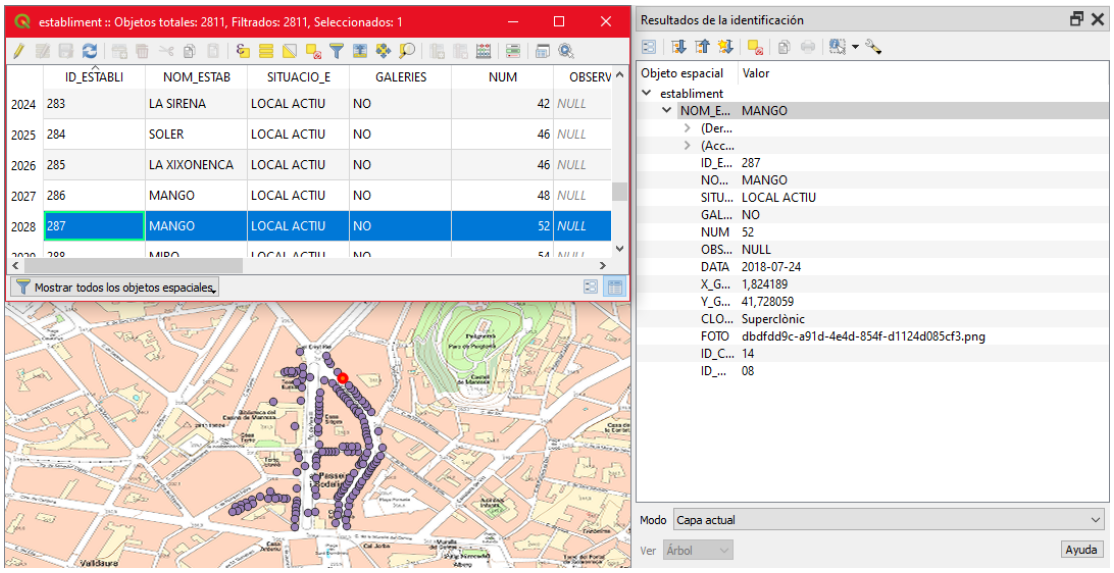


Ilustración 2- Captura de la tabla de datos, resultados de identificación y visualización en QGIS. Elaboración propia.

No obstante, este proceso solamente muestra la información almacenada en la tabla ESTABLIMENT. Para una visualización completa, que es lo que interesa en este caso, se crea una "vista" de las tres tablas a partir de las PK y FK (*Ilustración 7).*

3.2 Visor web

3.2.1 Diseño funcional e interfaz

El diseño y las funciones del visor se establecieron junto a Solucions. Desde un principio ya se pidió un visor sencillo e intuitivo, por lo que solamente se tenían que definir ciertos detalles. Estos son, por ejemplo, la simbolización, información del Pop-up o el número de filtros.

Durante el proceso de elaboración se ha intentado ajustar a las demandas del cliente y, además, dando más opciones para mejorar sus funciones.

3.2.2 Análisis de requerimientos

3.2.2.1 Requerimientos funcionales

ID	REQ_F_001
Nombre	Visor
Descripción	El sistema mostrará un visor centrado en Cataluña y podrá interactuar con él.
Requisito padre	-

ID	REQ_F_001_001
Nombre	Comercios
Descripción	El visor debe mostrar al usuario los comercios analizados geolocalizados
Requisito padre	REQ_F_001

ID	REQ_F_001_002
Nombre	Pop-up
Descripción	El cliente al clicar sobre un comercio se le abrirá un Pop-up donde se le muestre la información de la base de datos.
Requisito padre	REQ_F_001

ID	REQ_F_001_003
Nombre	Interacción
Descripción	El usuario debe poder interactuar con el visor a partir de las funciones de zoom (acercarse/alejarse), un gestor de información y un filtro.
Requisito padre	REQ_F_001

ID	REQ_F_001_004
Nombre	Gestor de información
Descripción	El gestor de información será capaz de dar la posibilidad de escoger el mapa base.
Requisito padre	REQ_F_001

ID	REQ_F_001_004_001
Nombre	Mapa base
Descripción	El visor debe dar la posibilidad de cambiar el mapa base a través del gestor.
Requisito padre	REQ_F_001_004

ID	REQ_F_001_005
Nombre	Filtrar información
Descripción	El usuario podrá elegir qué comercio se representa según su categoría.
Requisito padre	REQ_F_001

ID	REQ_F_001_006
Nombre	Zoom
Descripción	El visor ofrecerá un botón para acercar o alejar el zoom del mapa.
Requisito padre	REQ_F_001

ID	REQ_F_001_007
Nombre	Fullscreen
Descripción	El visor ofrecerá un botón para hacer que el visor esté en pantalla completa.
Requisito padre	REQ_F_001

ID	REQ_F_001_008
Nombre	Gestión de datos
Descripción	Los administradores podrán gestionar y modificar la base de datos añadiendo o eliminando registros desde el QGIS.
Requisito padre	REQ_F_001

3.2.2.2 Requerimientos no funcionales

ID	REQ_NF_001
Nombre	Adaptabilidad
Descripción	El sistema debe adaptarse a las especificaciones del navegador o sistemas operativos.
Requisito padre	-
ID	REQ_NF_002
Nombre	Conexión internet
Descripción	El visor debe tener conexión a internet para su correcto funcionamiento
Requisito padre	-

ID	REQ_NF_003
Nombre	Sistema de Gestión de Base de Datos
Descripción	La base de datos tiene que estar conectada a QGIS
Requisito padre	-

ID	REQ_NF_003_001
Nombre	Modificación de datos
Descripción	La edición de datos se realizará a través de QGIS para que la información se muestre en el visor.
Requisito padre	REQ_NF_003

ID	REQ_NF_004
Nombre	Errores
Descripción	El sistema debe ofrecer un porcentaje de errores, respecto al total de acciones, inferior al 5%.
Requisito padre	-

ID	REQ_NF_005
Nombre	Manejo intuitivo
Descripción	El visor debe ser diseñado de manera fácil de manejar para cualquier tipo de usuario.
Requisito padre	-

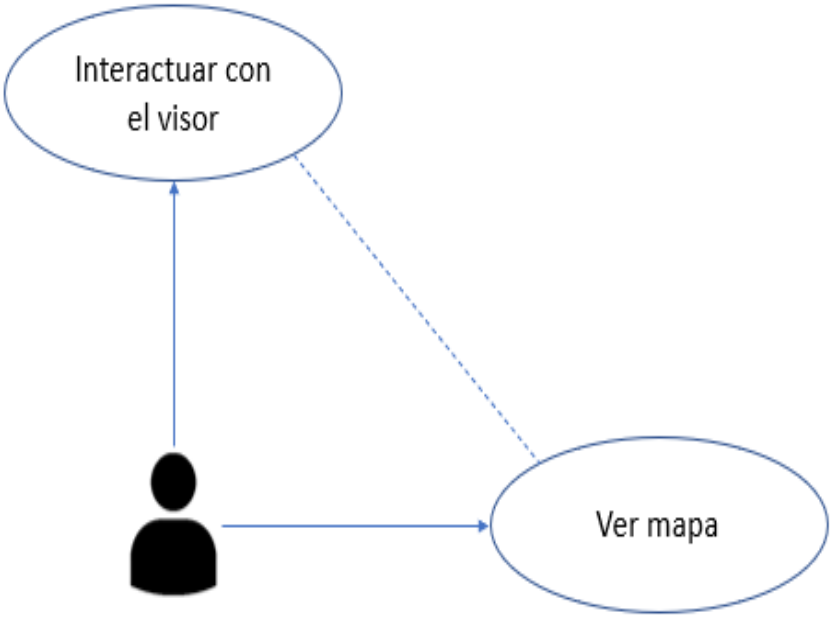
ID	REQ_NF_006
Nombre	Actualización y ampliación
Descripción	El sistema debe estar diseñado para posibles actualizaciones y ampliaciones futuras.
Requisito padre	-

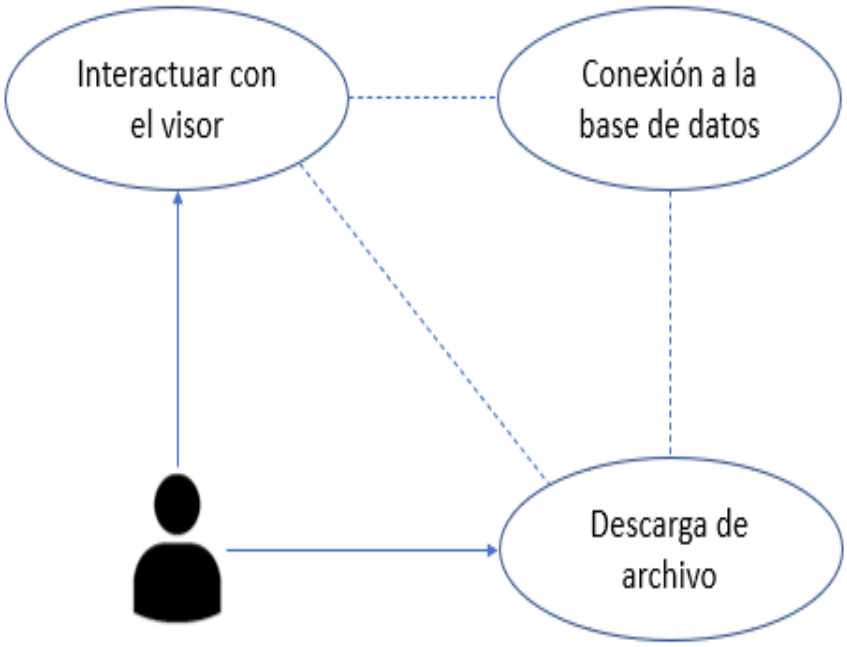
ID	REQ_NF_007
Nombre	Biblioteca y lenguajes de programación
Descripción	La biblioteca utilizada para los mapas donde se muestran los comercios será Leaflet.
Requisito padre	-

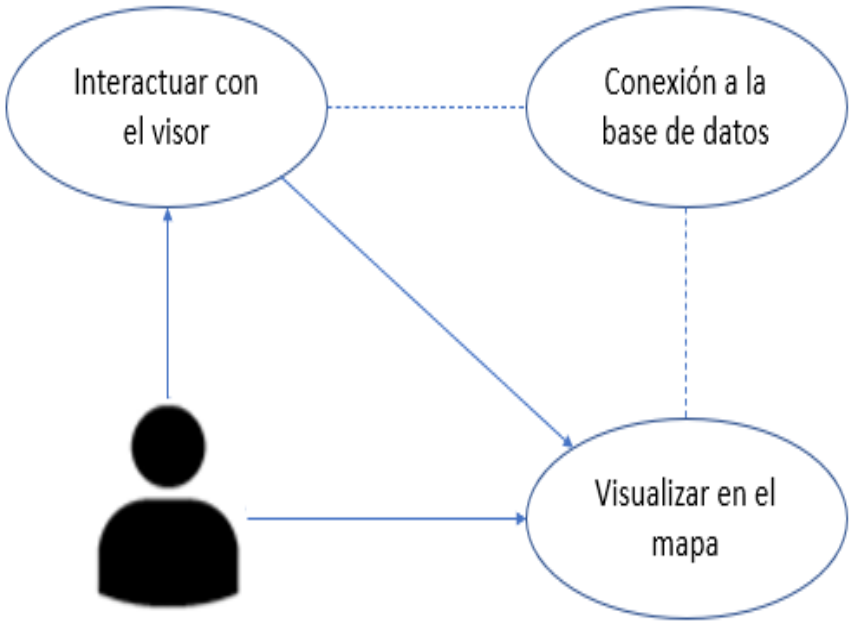
ID	REQ_NF_008
Nombre	Estilo
Descripción	El visor tendrá un estilo sencillo e intuitivo. Adaptado a las necesidades y demandas de la empresa.
Requisito padre	-

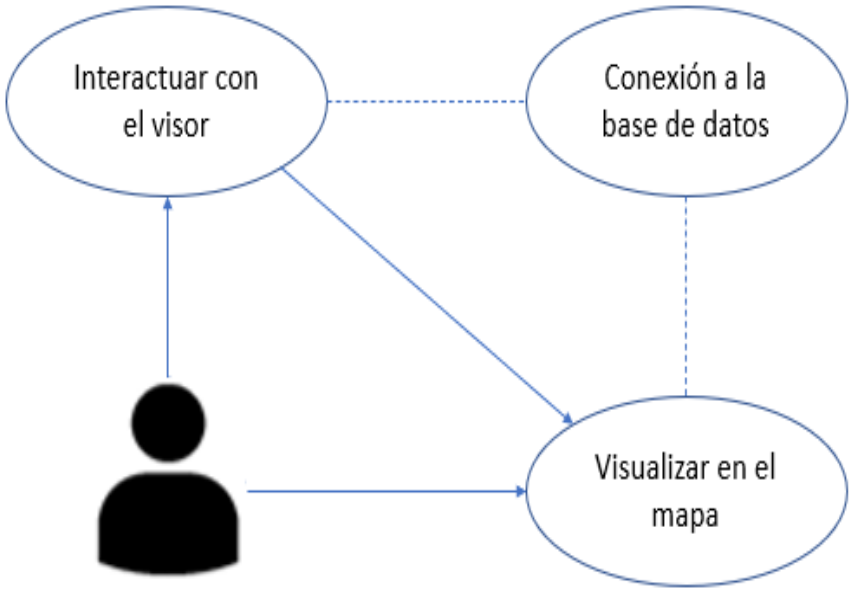
ID	REQ_NF_009
Nombre	Velocidad de carga
Descripción	El sistema tendrá una carga rápida en la aplicación de filtros, carga de mapas y pop-up.
Requisito padre	-

3.2.3 Casos de uso

Título	Explotación de información
Precondición	Conexión a internet. Que exista una agrupación por categorías de los comercios.
Descripción	En este caso se muestra como el usuario es capaz de filtrar la información y escoger el mapa según sus intereses. El mapa deberá mostrar todo lo que el usuario filtre.
Importancia	Alta
Prioridad	Alta
Diagrama	

Título	Descarga de información
Precondición	Conexión a internet. Que exista una agrupación por categorías de los comercios.
Descripción	En este caso se muestra como el usuario es capaz de descargar la información de los comercios. El sistema tendrá que devolver un archivo formato PDF o Excel.
Importancia	Alta
Prioridad	Alta
Diagrama	

Título	Consulta de información pop-up de comercios.
Precondición	Conexión a internet. Que exista una agrupación por categorías de los comercios.
Descripción	Aquí se representa como el usuario es capaz de clicar sobre uno de los comercios, cargar la información de éste en un pop-up y visualizarla.
Importancia	Alta
Prioridad	Alta
Diagrama	

Título	Búsqueda de comercios
Precondición	Conexión a internet. Que exista una agrupación por categorías de los comercios.
Descripción	El caso de uso en la búsqueda de comercios es posible a través el buscador del visor, en el cuál se podrá buscar según su categoría.
Importancia	Alta
Prioridad	Alta
Diagrama	

3.2.4 Solución metodológica de programación

Como ya se ha mencionado anteriormente, al ser un visor sencillo el número de funciones son básicas y simples.

Desde un principio se pensó en utilizar la librería Leaflet para el diseño de mapas. Esta, además de ser fácil de utilizar, es la que se ha utilizado para elaborar diferentes mapas interactivos a lo largo del máster.

Una vez descargado todos los archivos necesarios para el correcto funcionamiento del visor, se comenzó a generar el script. Los lenguajes utilizados son:

- HTML: Definir el script del visor y los plugin.
- JavaScript: Aplicar una mejor interactividad con el usuario del visor.
- CSS: Definir el estilo del sistema.

Ya que el visor solamente constará de una interfaz, simplemente se tienen que agregar los directorios a los archivos de Leaflet y generar un *script* para visualizar las diferentes capas del mapa. En este caso se definió una base en escala de grises, con tal de no aportar simbolización extra y quitar importancia a la posterior representación de los comercios.

La aparición de los comercios en el visor se realiza a través de un *geoJSON*, extraído de los datos en la base de datos, a partir de *QGIS.* Esto es debido a las características del servidor web de la empresa, que hacen imposible la conexión directa del visor web y la base de datos del servidor.

El pop-up de los comercios se ha definido a partir de unas variables de los comercios y una imagen de este, en el caso de existir.

A continuación, comenzó el proceso de instalación de los diferentes plugin: gestor de capas, *Fullscreen*, filtros y búsqueda.

- Gestor de capas: ofrece la posibilidad de escoger qué mapa base se quiere visualizar junto a los comercios. Por defecto, muestra un mapa en escala de grises, pero se ha añadido el *WMS* de ortoimagenes del ICGC, un mapa físico de *Stamen* y otro mapa base de *OpenStreetMaps.*

- *Fullscreen*: clicando sobre este icono, el visor se amplia en toda la pantalla y mejora su visualización.
- Filtro: Desde un principio se quiso añadir un filtro por la categoría de situación empresarial para mejorar la funcionalidad. Sin embargo, la gran cantidad de variables ofrecía la posibilidad de agregar más filtros por diferentes categorías. Al final se añadieron 4 filtros: municipio, situación empresarial, clonismo y actividad comercial.
- Búsqueda: este plugin es capaz de buscar por una variable de los comercios. En este caso, se ha definido el nombre y la dirección del comercio como campo de búsqueda, debido a la posibilidad de que un comercio se repita en varios municipios.

Una vez instalados los plugin y diseñada una correcta visualización de los comercios, toca definir el aspecto del ambiente del visor.

Como se ha repetido en varias ocasiones, es un visor sencillo, por lo que la ornamentación seguirá el mismo camino. Solamente se han aplicado colores pálidos para no destacar y pequeños bordes a los filtros y visor.

4 Resultados

4.1 Base de datos

Como se ha explicado anteriormente, la creación de la base de datos cuenta con varias etapas (modelos y carga de datos) que se mostrarán a continuación.

4.1.1 Modelo conceptual y lógico

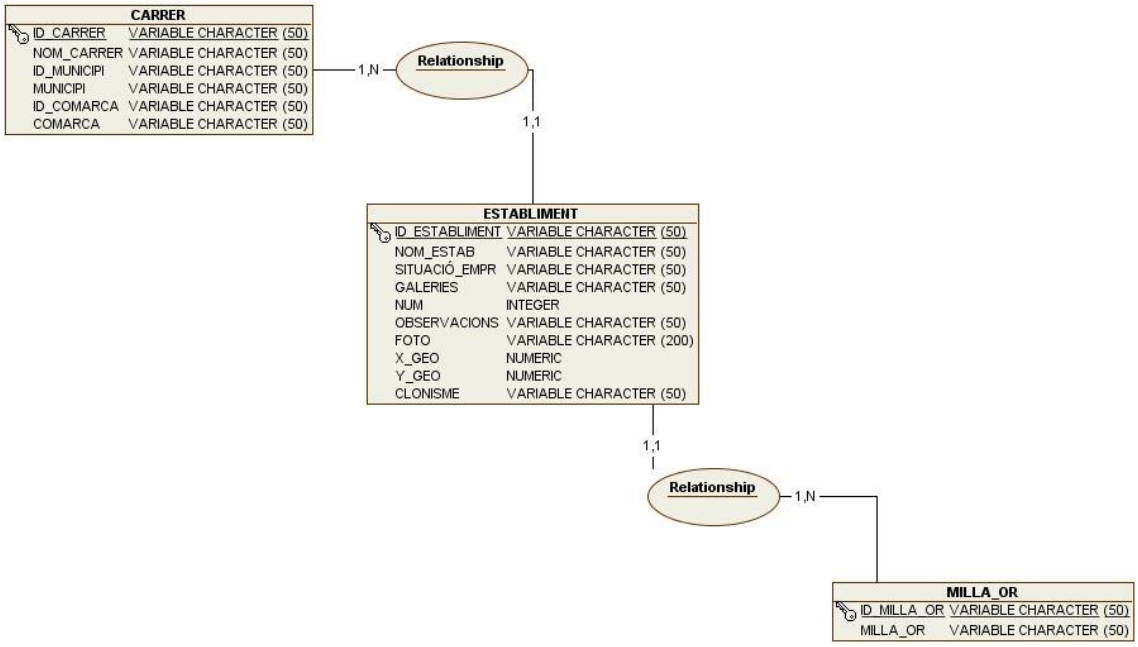


Ilustración 3- Modelo conceptual creado en OMS. Elaboración propia.

En la *ilustración 3* se muestra el modelo conceptual creado en *Open ModelSphere,* las diferentes tablas que lo forman y las relaciones.

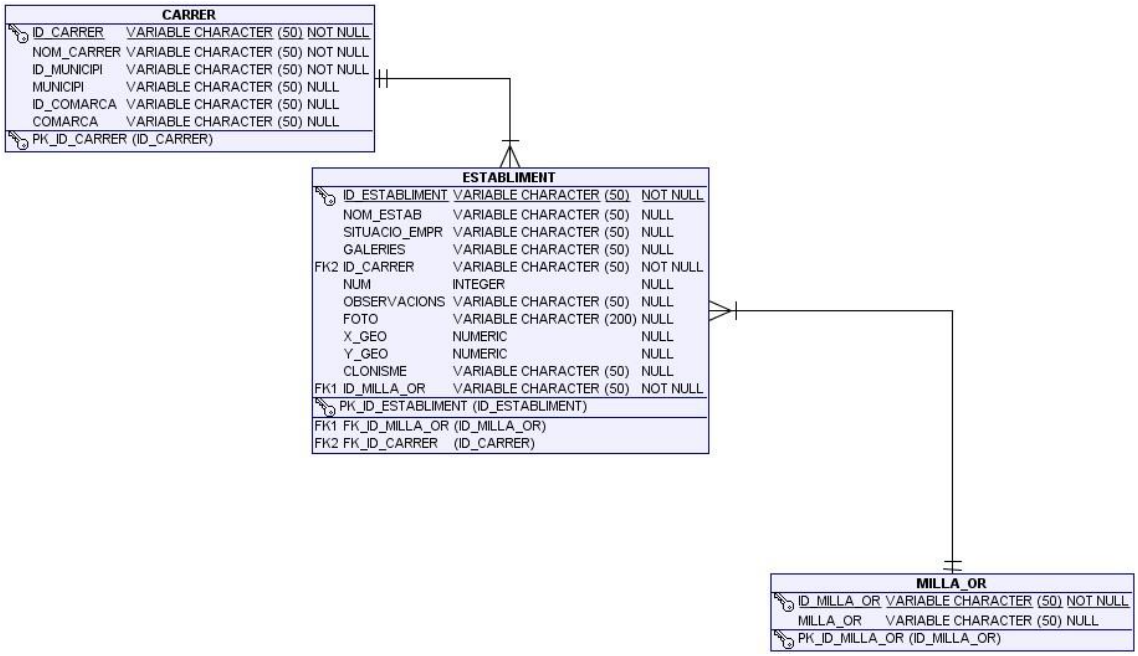


Ilustración 4- Modelo lógico creado en OMS. Elaboración propia.

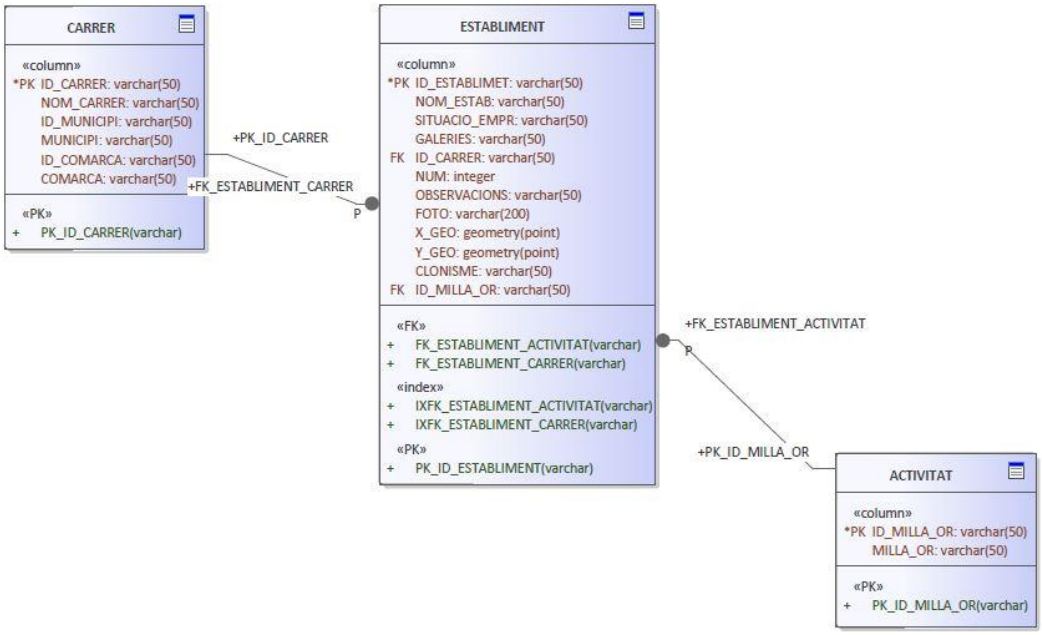


Ilustración 5- Modelo UML creado en EA. Elaboración propia.

En este caso se muestra el resultado del modelo UML en el programa *Enterprise Architech.* A partir de este se obtiene el *script SQL* para crear la estructura de la base de datos.

El código *SQL* de la estructura está disponible en el anexo 1.

4.1.2 Carga de datos

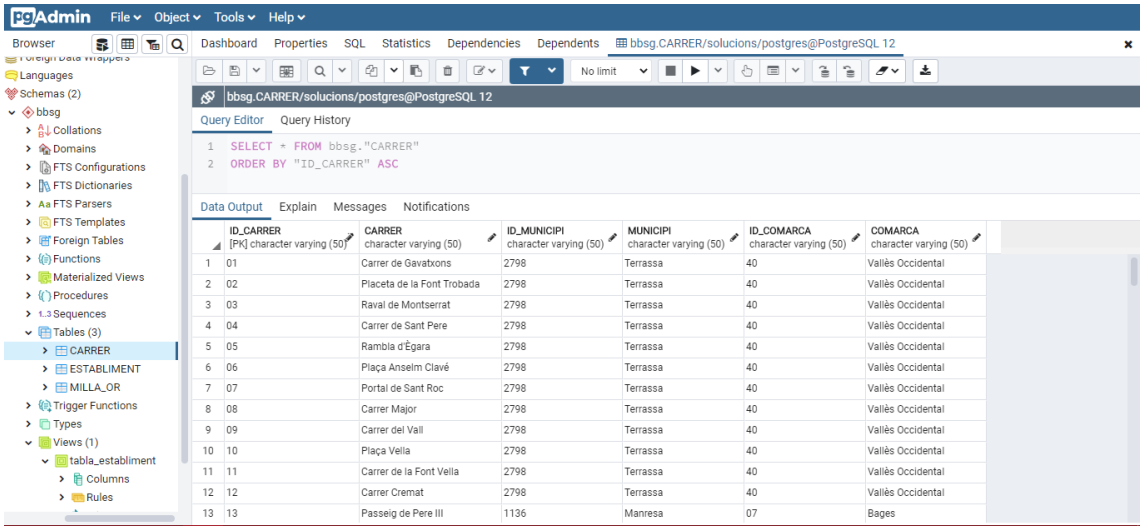


Ilustración 6- Carga de datos en PostgreSQL de la tabla CARRER. Elaboración propia.

En la ilustración 6 se observa una captura de pantalla donde se muestra parte de la tabla CARRER y los campos que la forman.

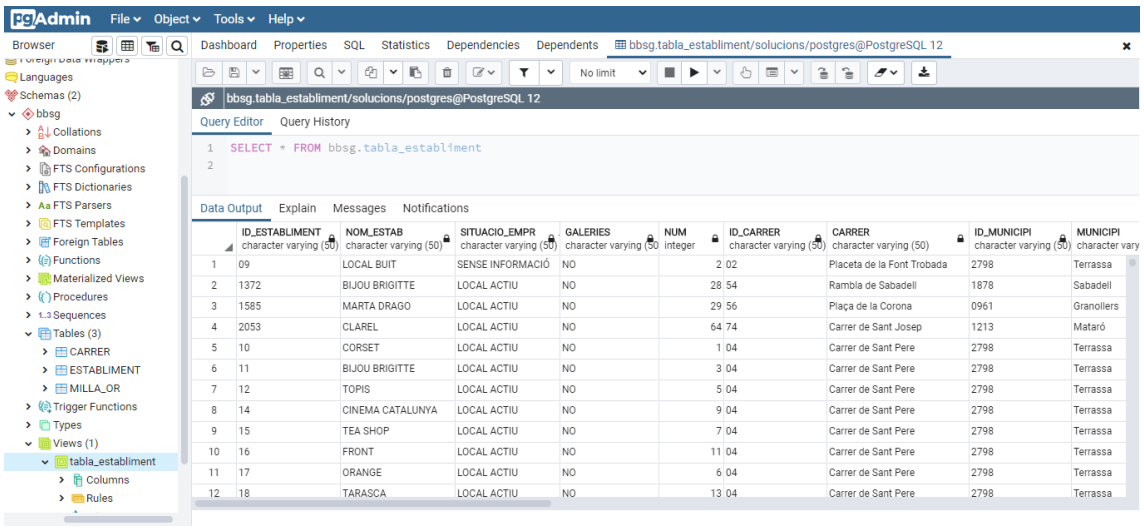


Ilustración 7- Carga completa de los datos en la vista TABLA_ESTABLIMENT en PostgreSQL. Elaboración propia.

4.2 Visor web

Por su parte, el visor web solamente ha tenido una parte de desarrollo a partir de su codificación. Sin embargo, la combinación de lenguaje HTML, JavaScript y CSS ha permitido crear un visor personalizado y con las funciones deseadas.

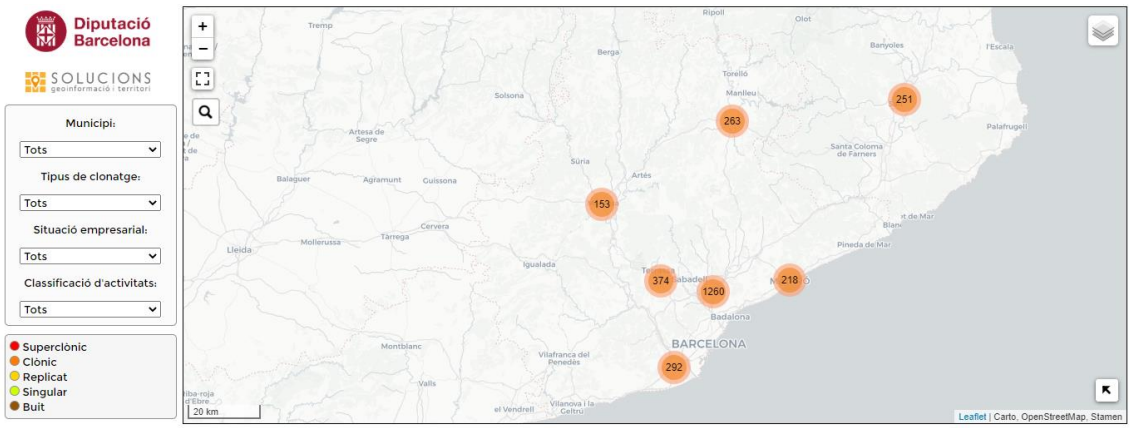


Ilustración 8- Visualización general del visor web. Elaboración propia.

Ilustración 9- Ejemplo de pop-up. Elaboración propia.

El pop-up de los comercios se ha personalizado de manera sencilla y discreta para no desentonar con el resto del visor.

5 Conclusiones

Realizar las prácticas del máster en Geoinformación en *Solucions Geogràfiques* ha aportado experiencia, conocimiento y motivación en partes iguales. Esto es debido a que se trata de una cooperativa del ámbito geográfico en constante crecimiento. Así mismo, ha servido de ayuda para obtener experiencia en la elaboración de proyectos profesionales y reales y, además, conocer una parte del mundo profesional del geógrafo.

Crear el visor web, en aspectos generales, no ha tenido un nivel elevado de complejidad. Ante la petición por parte de la empresa de que sea sencillo, el periodo de prácticas y las circunstancias de teletrabajo, se ha intentado adaptar al máximo las características del visor a las demandadas.

En una posible futura actualización del visor, hay diferentes temas que se podrían mejorar, como, por ejemplo, el aspecto o las funcionalidades. En el primer caso, se podría diseñar una mejor distribución de los filtros, los logos o los mapas. En cambio, en el segundo se podrían añadir más funcionalidades, como, por ejemplo, filtros, una leyenda variable o la opción de poder descargar una ficha personalizada de cada comercio o todos los datos para hacer estudios individuales. Otra actualización, sería terminar de conectar el visor web a la base de datos del servidor. En este caso no ha sido posible debido a la configuración del servidor de la empresa y el breve tiempo de las prácticas.

En definitiva, realizar este proyecto ha sido una experiencia enriquecedora en el ámbito académico y profesional.

6 Referencias

Memoria:

- Eva Sivillà (2018). *Millora del visor d'espais VAT.*
- Kevin González (2018). *Creació d'un visor web per la visualització de dades d'interferometria de l'Institut Cartogràfic i Geològic de Catalunya.*
- Pau Estruch (2019). *Creació d'una xarxa de rutes ciclista: Topologia, visualització i càlcul dels camins òptims.*
- TIC Mallorca (2020). Prototipo del *Visor d'establiments oberts durant el confinement.* Disponible en: https://www.google.com/maps/d/u/0/viewer?mid=1VPcIA0Bz1TYxsGFQaHivjHugK2Kghxq3&ll=39.70582390000005%2C2.7901382999999846&z=9 [Consulta: Junio 2020]

Visor web:

- Leaflet: https://leafletjs.com/ [Consulta: junio 2020]
- Leaflet-plugin: https://leafletjs.com/plugins.html [Consulta: junio 2020]
- Mapa base escala grises: https://carto.com/ [Consulta: junio 2020]
- Mapa base OpenStreetMaps: https://www.openstreetmap.org [Consulta: junio 2020]
- Mapa físico: https://stamen.com/ [Consulta: junio 2020]
- MappingGis: https://mappinggis.com/ [Consulta: junio 2020]
- w3schools: https://www.w3schools.com/ [Consulta: Junio 2020]
- WMS ICGC ortoimagenes: https://www.icgc.cat/ca/ [Consulta: junio 2020]

7 Anexos

Anexo 1 - Script completo de la estructura de la base de datos:

```
/* ---------------------------------------------------- */
/* Generated by Enterprise Architect Version 15.0       */
/* Created On: 09-may.-2020 12:41:11                    */
/* DBMS      : PostgreSQL                               */
/* ---------------------------------------------------- */

/* Drop Tables */

DROP TABLE IF EXISTS "ACTIVITAT" CASCADE
;

DROP TABLE IF EXISTS "CARRER" CASCADE
;

DROP TABLE IF EXISTS "ESTABLIMENT" CASCADE
;

/* Create Tables */

CREATE TABLE "ACTIVITAT"
(
	"ID_MILLA_OR" varchar(50) NOT NULL,
	"MILLA_OR" varchar(50) NULL
)
;

CREATE TABLE "CARRER"
(
	"ID_CARRER" varchar(50) NOT NULL,
	"NOM_CARRER" varchar(50) NULL,
	"ID_MUNICIPI" varchar(50) NULL,
	"MUNICIPI" varchar(50) NULL,
	"ID_COMARCA" varchar(50) NULL,
	"COMARCA" varchar(50) NULL
)
;

CREATE TABLE "ESTABLIMENT"
(
	"ID_ESTABLIMET" varchar(50) NOT NULL,
	"NOM_ESTAB" varchar(50) NULL,
	"SITUACIO_EMPR" varchar(50) NULL,
	"GALERIES" varchar(50) NULL,
	"ID_CARRER" varchar(50) NULL,
	"NUM" integer NULL,
	"OBSERVACIONS" varchar(50) NULL,
	"FOTO" varchar(50) NULL,
	"X_GEO" geometry(point) NULL,
	"Y_GEO" geometry(point) NULL,
	"CLONISME" varchar(50) NULL,
	"ID_MILLA_OR" varchar(50) NULL
)
;

/* Create Primary Keys, Indexes, Uniques, Checks */

ALTER TABLE "ACTIVITAT" ADD CONSTRAINT "PK_ID_MILLA_OR"
	PRIMARY KEY ("ID_MILLA_OR")
;

ALTER TABLE "CARRER" ADD CONSTRAINT "PK_ID_CARRER"
	PRIMARY KEY ("ID_CARRER")
```

```
;

ALTER TABLE "ESTABLIMENT" ADD CONSTRAINT "PK_ID_ESTABLIMENT"
	PRIMARY KEY ("ID_ESTABLIMET")
;

CREATE INDEX "IXFK_ESTABLIMENT_ACTIVITAT" ON "ESTABLIMENT" ("ID_MILLA_OR" ASC)
;

CREATE INDEX "IXFK_ESTABLIMENT_CARRER" ON "ESTABLIMENT" ("ID_CARRER" ASC)
;

/* Create Foreign Key Constraints */

ALTER TABLE "ESTABLIMENT" ADD CONSTRAINT "FK_ESTABLIMENT_ACTIVITAT"
	FOREIGN KEY ("ID_MILLA_OR") REFERENCES "ACTIVITAT" ("ID_MILLA_OR") ON DELETE No
Action ON UPDATE No Action
;

ALTER TABLE "ESTABLIMENT" ADD CONSTRAINT "FK_ESTABLIMENT_CARRER"
	FOREIGN KEY ("ID_CARRER") REFERENCES "CARRER" ("ID_CARRER") ON DELETE No
Action ON UPDATE No Action
;
```